AMENDMENTS TO THE CLAIMS:

1. (Currently amended) A method for determining the correctness of a potential interprocedural dead store optimization for an optimizing compiler, the optimizing compiler generating an intermediate representation of code to be compiled, including a call graph, the method comprising a top-down traversal of the call graph, and, for each procedure definition reached in the call graph traversal, further comprising:

determining a live on exit set of global variables for the reached procedure definition;

determining a live on exit set of global variables for each procedure call point within the reached procedure definition;

storing the determined live on exit set of global variables for each procedure call point in a live on exit data structure; and

using the determined live on exit set of global variables for the reached procedure definition to determine global variables that are ineligible for interprocedural dead store elimination in the reached procedure definition.

2. (Currently amended) The method of claim 1 in which the live on exit set of global variables for the reached procedure definition is determined by taking the union of all stored entries in the live on exit data structure corresponding to call points for the reached procedure.

3. (Currently amended) The method of claim 2 in which determining the live on exit set of global variables for each procedure call point within the reached procedure definition

comprises:

determining a basic block live set for each block of computer code in a control flow graph for the reached procedure definition, the basic block live set comprising the variables used in the block of computer code and the variables used in any procedure called within the block of computer code; and

determining the live on exit set of global variables for each procedure call point by taking the union of the basic block live sets for all successor blocks to the block in the control flow graph containing the procedure call point, and adjusting the union to include uses of variables in the code between the call point for the procedure and the end of the block containing the call point.

4. (Currently amended) The method of claim 1 in which determining the live on exit set for each procedure call point in the reached procedure definition comprises:

determining a basic block live set for each block of computer code in a control flow graph for the reached procedure definition, the basic block live set comprising the variables used in the block of computer code and the variables used in any procedure called within the block of computer code; and

determining the live on exit set for each procedure call ~~by~~ point by taking the union of the basic block live sets for all successor blocks to the block in the control flow graph containing the procedure call point, and adjusting the union to include uses of variables in the code between the call point for the procedure and the end of the block containing the call point.

5. (Currently amended) The method of claim 2, further comprising, after determining the live on exit set of global variables for the reached procedure definition, removing all entries in the live on exit data structure corresponding to call points for the reached procedure.

6. (Original) The method of claim 3, in which the variables used in a procedure called within a block of computer code are determined by accessing the mod/use set for the procedure associated with the procedure definition node in the call graph.

7. (Currently amended) The method of claim 1 in which using the live on exit set of global variables for the reached procedure definition to determine the variables that are ineligible for interprocedural dead store elimination in the reached procedure definition comprises generating pseudo uses of the members of the live on exit set of global variables for the reached procedure definition in the data flow graph for the reached procedure definition.

8. (Previously presented) The method of claim 1 in which the live on exit data structure comprises bit vector entries and is indexed by call graph edges.

9. (Currently amended) The method of claim 2 further comprising using the live on exit set of global variables for the reached procedure definition to determine whether the

reached procedure definition may be cloned by the optimizing compiler.

10. (Currently amended) A method for determining the correctness of a potential interprocedural dead store optimization for an optimizing compiler, the optimizing compiler generating an intermediate representation of code to be compiled, including a call graph, the method comprising a top-down traversal of the call graph, and, for each procedure definition reached in the call graph traversal, further comprising:

______determining a live on exit set of global variables for each procedure call point within the reached procedure definition by:

______determining a basic block live set for each block of computer code in a control flow graph for the reached procedure definition, the basic block live set comprising the global variables used in the block of computer code and the global variables used in any procedure called within the block of computer code; and

determining the live on exit set of global variables for each procedure call point by taking the union of the basic block live sets for all successor blocks to the block in the control flow graph containing the procedure call point, and adjusting the union to include uses of global variables in the code between the call point for the procedure and the end of the block containing the call point;

storing the determined live on exit set of global variables for each procedure call point in a live on exit data structure comprising a bit vector indexed by a call graph edge;

determining a live on exit set of global variables for the reached procedure definition by taking the union of all stored entries in the live on exit data structure corresponding to call points for the reached procedure;

removing all entries in the live on exit data structure corresponding to call points for the reached procedure; and

using the live on exit set of global variables for the reached procedure definition to determine global variables that are ineligible for interprocedural dead store elimination in the reached procedure definition.

11. (Previously presented) A computer program product for the compilation of computer code, the computer program product comprising a computer usable medium having computer readable code means embodied in said medium, comprising computer readable program code means to carry out the method of claim 1.

12. (Currently amended) An optimizing compiler comprising:

means for generating an intermediate representation of computer code, the intermediate representation comprising a call graph;

means for traversing the call graph in top down order;

means for storing a live on exit data structure;

means for generating a record in the live on exit data structure for each procedure call encountered in the traversal of the call graph, the record comprising data representing global variables that are live at the point of the procedure call; and

means for calculating the live on exit set for a procedure definition reached in traversing the call graph,

wherein the means for calculating the live on exit set ~~comprising~~ comprises means for retrieving records from the live on exit data structure corresponding to the reached procedure definition, means for performing a union of the records to determine the live on exit set for the reached procedure definition, and means for signaling the availability of a dead store elimination optimization for a store operation contained in the reached procedure definition based on the live on exit set calculated for the procedure definition.

13. (Original) The optimizing compiler of claim 12, further comprising means for removing records associated with the reached procedure definition from the live on exit data structure following calculation of the live on exit set for the reached procedure definition.

14. (Currently amended) A component for determining the correctness of a potential interprocedural dead store optimization for an optimizing compiler, the optimizing compiler generating an intermediate representation of code to be compiled, including a call graph, the component comprising:

means to traverse the call graph in top-down order,

means for determining a live on exit set of global variables for each procedure call point within a reached procedure definition during the call graph traversal by:

~~   ~~determining a basic block live set for each block of computer code in a control flow graph for the reached procedure definition, the basic block live set comprising the global variables used in the block of computer code and the global variables used in any procedure called within the block of computer code, and

~~   ~~determining the live on exit set for each procedure call by taking the union of the basic block live sets for all successor blocks to the block in the control flow graph containing the procedure call point and by adjusting the union to include uses of global variables in the code between the call point for the procedure and the end of the block containing the call point,

means for storing the live on exit set of global variables for each procedure call point in an entry in a live on exit data structure comprising a bit vector indexed by a call graph edge,

means for determining a live on exit set of global variables for the reached procedure determination by taking the union of all stored entries in the live on exit data structure corresponding to call points for the reached procedure,

means for removing all entries in the live on exit data structure corresponding to call points for the reached procedure following determination of the live on exit set of global variables for the reached procedure definition, and

means for determining the global variables that are ineligible for interprocedural dead store elimination in the reached procedure definition, using the live on exit set of global variables for the reached procedure definition.

15. (Currently amended) The method of claim 1, further comprising determining variables, other than ~~those~~ variables determined to be ineligible for interprocedural dead store elimination, to be eligible for interprocedural dead store elimination.

16. (Currently amended) The method of claim 15, further comprising eliminating ~~those~~ the variables determined to be eligible for interprocedural dead store elimination.

17. (Currently amended) The method of claim 10, further comprising determining variables, other than ~~those~~ variables determined to be ineligible for interprocedural dead store elimination, to be eligible for interprocedural dead store elimination.

18. (Currently amended) The method of claim 17, further comprising eliminating ~~those~~ the variables determined to be eligible for interprocedural dead store elimination.